

Tarea Evaluable Docker -

2EV_Ejercicio_3_Imagen_con_Dockerfile

Módulo: Despliegue de aplicaciones Web - Distancia

Alumno: Alfonso Dapena Cores

Tarea Evaluable Docker - 2EV_Ejercicio_3_Imagen_con_Dockerfile

3.1 Consideraciones previas

3.1 Creación fichero `Dockerfile`.

3.3 Creación y ejecución del contenedor.

3.4 Subimos la imagen a nuestro repositorio en `Docker Hub`

3.5 Borramos la imagen

3.6 Creación nuevo contenedor modificando el puerto

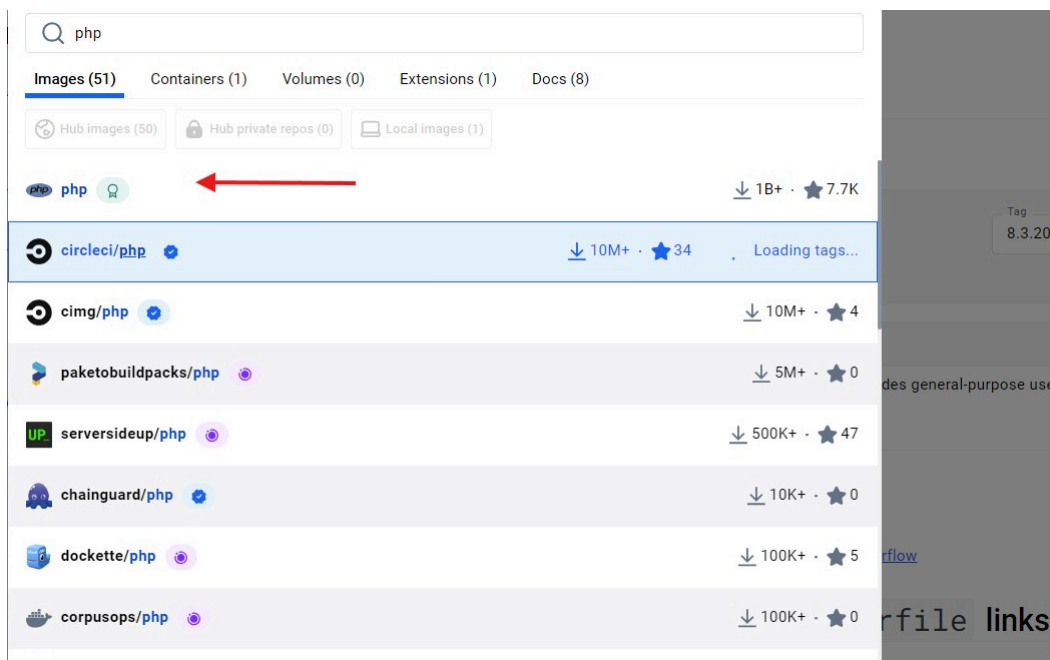
3.1 Consideraciones previas

- El archivo `Dockerfile` es un archivo de texto que contiene las instrucciones precisas para crear una imagen de Docker. En este fichero se define el entorno, las dependencias y configuraciones necesarias para que los contenedores funcionen correctamente. Dentro de un fichero `Dockerfile` nos podemos encontrar con varias instrucciones, siendo las mas comunes:

1. `FROM`: Identifica la imagen base.
2. `RUN`: Permite ejecutar comandos en el contenedor.
3. `WORKDIR`: establece el directorio de trabajo para ejecutar las instrucciones que siguen.
4. `ENV`: Establece variables de entorno para el contenedor.
5. `COPY`: Copia archivos y directorios desde el sistema host a la imagen de Docker.
6. `CMD`: Es comando que se ejecuta por defecto al iniciar un contenedor de Docker.
7. `ENTRYPOINT`: Define el comando y los parámetros que se ejecutan primero cuando se ejecuta el contenedor.
8. `EXPOSE`: indica los puertos en los que escuchará un contenedor.

3.1 Creación fichero `Dockerfile`.

- En primer lugar, antes de preparar el escenario donde vamos a ejecutar nuestro fichero `Dockerfile`, debemos explorar la imagen `PHP`, en nuestra aplicación Docker Desktop:



- En la documentación, nos encontraremos un ejemplo del fichero `Dockerfile` que vamos a ejecutar.

php

Docker Official Image 1B+ · 7.7K [View on Hub](#)

How to use this image

Create a `Dockerfile` in your PHP project

```
FROM php:8.2-cli
COPY . /usr/src/myapp
WORKDIR /usr/src/myapp
CMD [ "php", "./your-script.php" ]
```

Then, run the commands to build and run the Docker image:

```
$ docker build -t my-php-app .
$ docker run -it --rm --name my-running-app my-php-app
```

- A continuación, debemos de preparar el entorno donde ejecutaremos nuestro fichero `Dockerfile`. Para ello crearemos los directorios y archivos necesarios para nuestra aplicación con las siguientes líneas de comandos:

```
$ mkdir app #creamos el directorio que incluye los ficheros que servirá la aplicación
$ touch Dockerfile #creamos el fichero Dockerfile
```

```
adcorescliente@alfonsocliente: ~/ejercicio3
adcorescliente@alfonsocliente:~/ejercicio3$ mkdir app
adcorescliente@alfonsocliente:~/ejercicio3$ touch Dockerfile
adcorescliente@alfonsocliente:~/ejercicio3$ ls
app  Dockerfile
adcorescliente@alfonsocliente:~/ejercicio3$
```

```
adcorescliente@alfonsocliente:~/ejercicio3/app$ ls
estilo.css  fecha.php  index.html
adcorescliente@alfonsocliente:~/ejercicio3/app$
```

- El siguiente paso consistiría en editar el fichero `Dockerfile` y añadir las instrucciones que se ejecutarán cuando construyamos nuestra imagen.

```
#syntax=docker/dockerfile:1
FROM php:7.4-apache #seleccionamos la imagen base con la que construiremos la
nuestra
COPY app /var/www/html #copiamos los archivos del directorio app al root de
Apache
EXPOSE 80 #nos indica el puerto a través del que se escucharán las peticiones
```

```
GNU nano 6.2 Dockerfile
# syntax=docker/dockerfile:1
FROM php:7.4-apache
COPY app /var/www/html/
EXPOSE 80
```

- Una vez creado nuestro fichero `Dockerfile`, procedemos a construir nuestra imagen. Para que podamos subirla con posterioridad a `Docker Hub`, indicaremos nuestro usuario en `Docker Hub`, alfonsoadapenacores73, seguido del nombre de la imagen, ejercicio3, y su etiqueta, v3.

```
$ docker build -t alfonsoadapenacores73/ejercicio3:v3
```

```
adcorescliente@alfonsocliente:~/ejercicio3$ docker build -t alfonsoadapenacores73/ejercicio3:v3 .
[+] Building 5.6s (11/11) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 121B                               0.0s
=> resolve image config for docker-image://docker.io/docker/dockerfile:1 3.1s
=> [auth] docker/dockerfile:pull token for registry-1.docker.io 0.0s
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:4c68376a702446fc3c79af22de146a148bc3367e73c25a5803d453b6b 0.0s
=> [internal] load metadata for docker.io/library/php:7.4-apache 1.0s
=> [auth] library/php:pull token for registry-1.docker.io         0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [internal] load build context                                  0.0s
=> => transferring context: 125B                                    0.0s
=> [1/2] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d 0.0s
=> CACHED [2/2] COPY app /var/www/html/                          0.0s
=> exporting to image                                             0.0s
=> exporting layers                                              0.0s
=> writing image sha256:af8b68ef914e97f212131813d1db6472c3a557c560687ca612d34136b40d46cf 0.0s
=> naming to docker.io/alfonsoadapenacores73/ejercicio3:v3      0.0s
```

3.3 Creación y ejecución del contenedor.

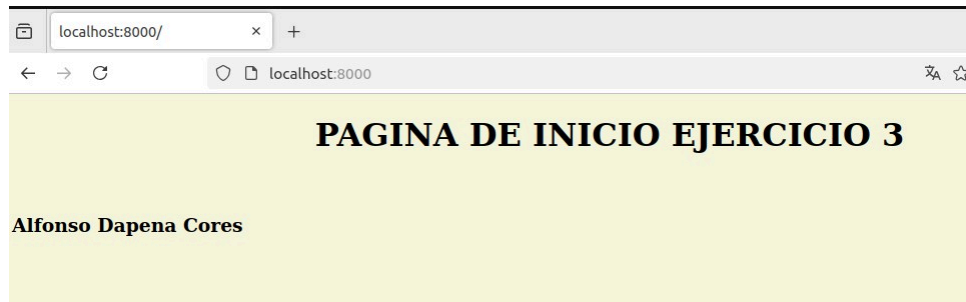
- Ahora que ya tenemos nuestra imagen lista, procedemos a la creación de nuestro contenedor, al que llamaremos ejercicio_3. A continuación comprobaremos que el contenedor se está ejecutando

```
$ docker run -d -p 8000:80 --name ejercicio_3
alfonsoadapenacores73/ejercicio3:v3
$ docker ps
```

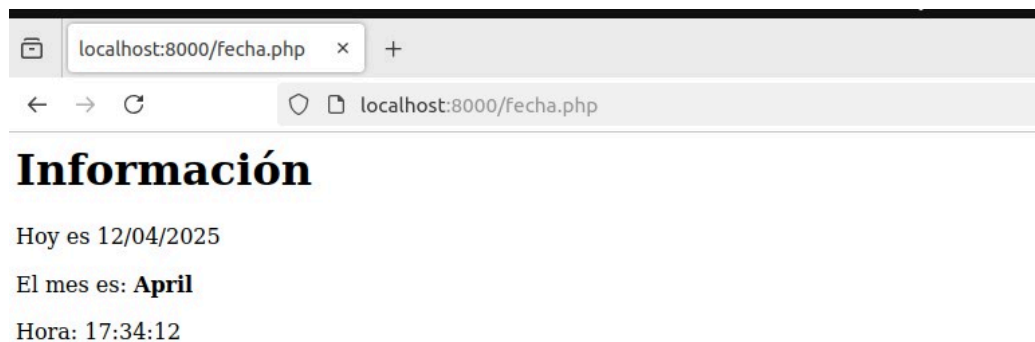
```
adcorescliente@alfonsocliente:~/ejercicio3$ docker run -d -p 8000:80 --name ejercicio_3 alfonsoadapenacores73/ejercicio3:v3
619cc534ce4f71fac0ecd27c76f198e843f27f8b91149d947eb1b0683e3b5ae0

adcorescliente@alfonsocliente:~/ejercicio3$ docker ps
CONTAINER ID   IMAGE                                NAMES      COMMAND                  CREATED        STATUS        PORTS
619cc534ce4f   alfonsoadapenacores73/ejercicio3:v3  ejercicio_3 "docker-php-entrypoi..." 6 minutes ago Up 6 minutes   0.0.0.0:8000->80/tcp
```

- Lo siguiente será comprobar en nuestro navegador que podemos visualizar la página `index.html` de nuestro servidor apache. Para ello introducimos la siguiente dirección en nuestro navegador: <http://localhost:8000>



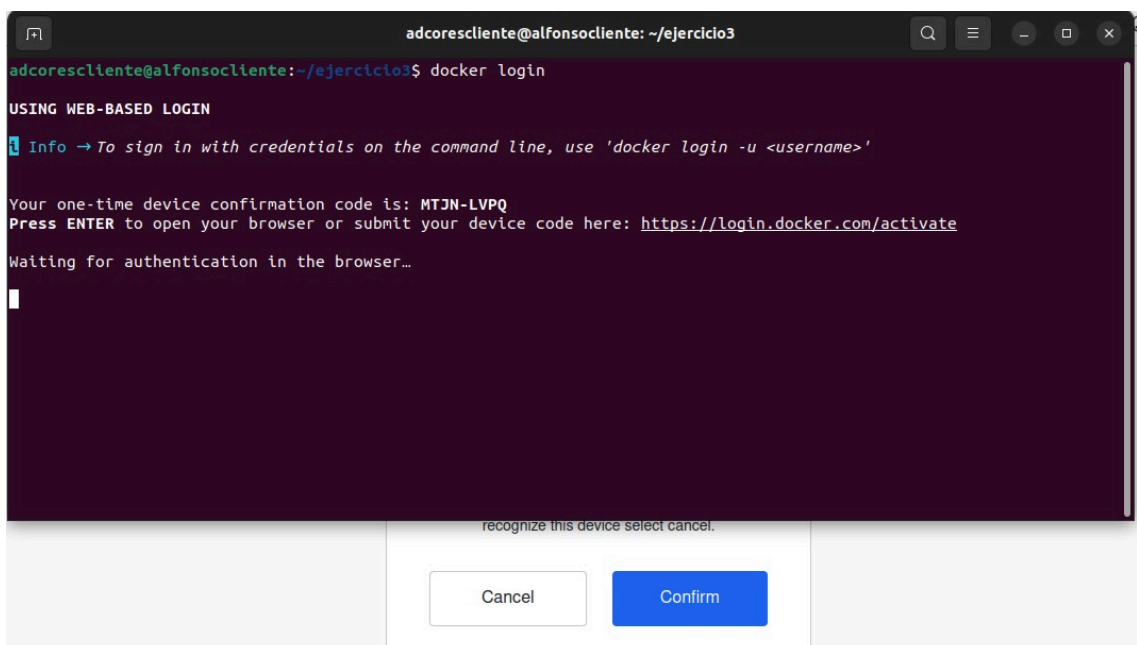
- A continuación comprobamos que el script `php` funciona correctamente introduciendo la siguiente dirección en nuestro navegador : <http://localhost:8000/fecha.php>

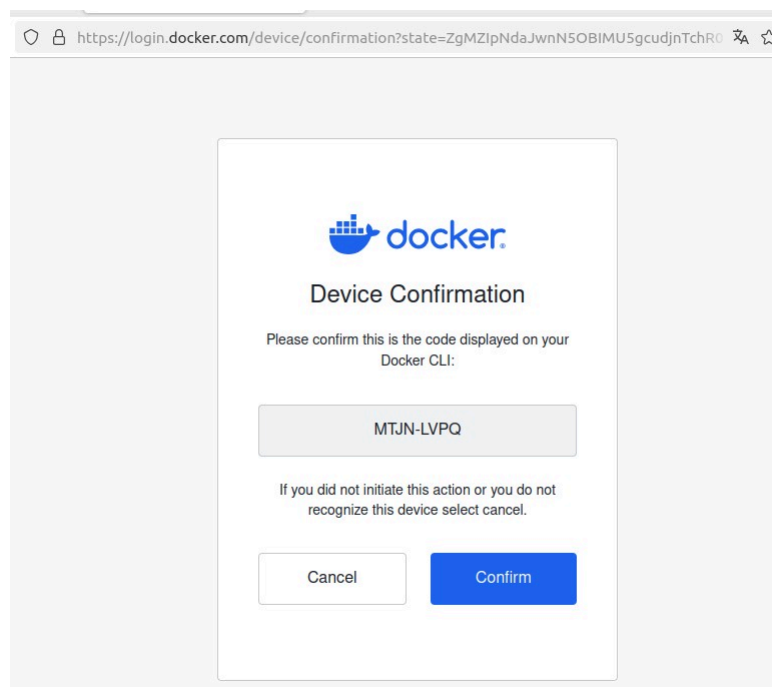


3.4 Subimos la imagen a nuestro repositorio en Docker Hub

- Para subir nuestra imagen a nuestro repositorio en `Docker Hub`, primero debemos logearnos. Si es la primera vez que lo hacemos, se nos mostrará un enlace que debemos pinchar para autenticarnos:

```
$ docker login
```





- Después de logearnos con éxito se nos mostrará en pantalla el siguiente mensaje:

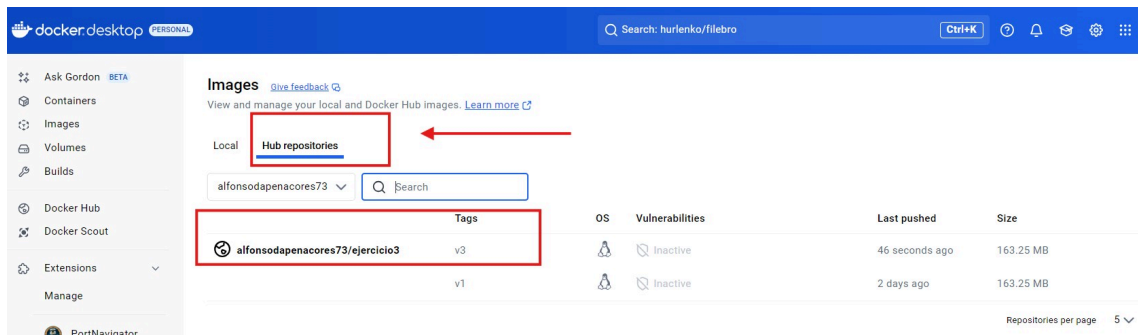
```
adcorescliente@alfonsocliente: ~  
adcorescliente@alfonsocliente:~$ docker login  
Authenticating with existing credentials... [Username: alfonsodapenacores73]  
  
Info → To login with a different account, run 'docker logout' followed by 'docker login'  
  
Login Succeeded  
adcorescliente@alfonsocliente:~$
```

- Ahora que ya nos hemos logeado, deberemos de subir la imagen a nuestro repositorio. Para ello introduciremos el siguiente código:

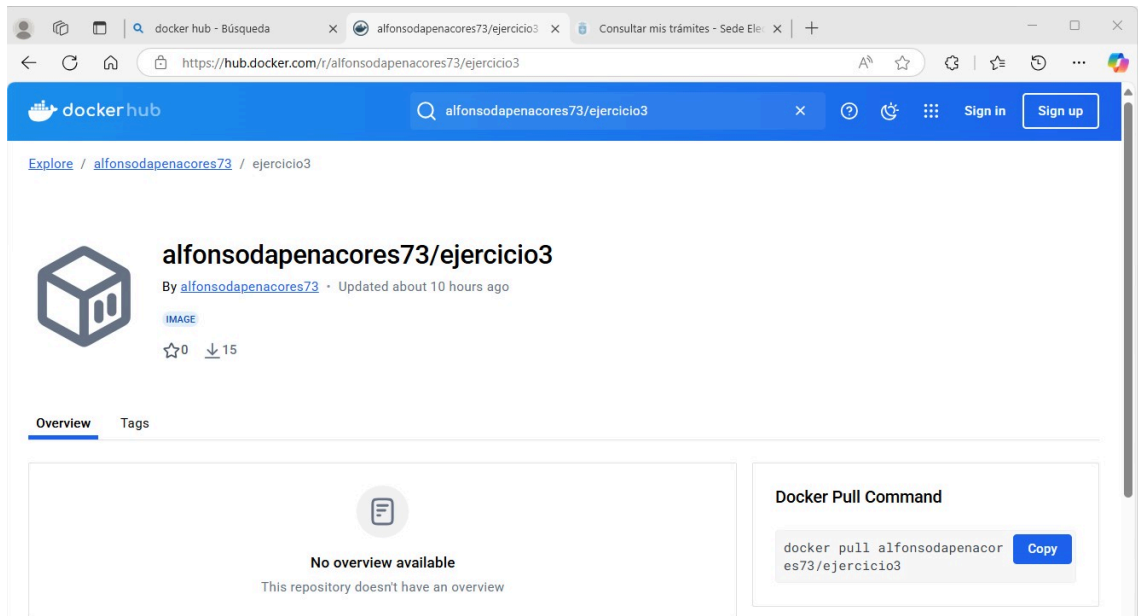
```
$docker image push alfonsodapenacores73/ejercicio3:v3
```

```
adcorescliente@alfonsocliente: ~/ejercicio3  
adcorescliente@alfonsocliente:~/ejercicio3$ docker image push alfonsodapenacores73/ejercicio3:v3  
The push refers to repository [docker.io/alfonsodapenacores73/ejercicio3]  
05e4b1d96be7: Layer already exists  
8d33242bf117: Layer already exists  
529016396883: Layer already exists  
5464bcc3f1c2: Layer already exists  
28192e867e79: Layer already exists  
d173e78df32e: Layer already exists  
0be1ec4fbfdc: Layer already exists  
80fa0c430434: Layer already exists  
a538c5a6e4e0: Layer already exists  
e5d40f64dcb4: Layer already exists  
44148371c697: Layer already exists  
797a7c0590e0: Layer already exists  
f60117696410: Layer already exists  
ec4a38999118: Layer already exists  
v3: digest: sha256:58b5d43fc9ab63785a08fd1011ff2234a7a06105eeef18b3e5b0e9f83587c44a size: 3242  
adcorescliente@alfonsocliente:~/ejercicio3$
```

- Accedemos a nuestra aplicación **Docker Desktop** y comprobamos que nuestra imagen se ha subido correctamente:



- También podemos comprobar en **Docker Hub**, la existencia de la imagen:



3.5 Borramos la imagen

- Una vez hemos comprobado que nuestro contenedor se ejecuta correctamente y que hemos subido nuestra imagen a nuestro repositorio remoto, procedemos a borrar la misma en nuestro repositorio local:

```
$ docker rmi alfonsodapenacores73/ejercicio3:v3 // borramos la imagen
$ docker images // comprobamos que la misma no figure en nuestro repositorio local
```



```
adcorescliente@alfonsocliente: ~/ejercicio3
adcorescliente@alfonsocliente:~/ejercicio3$ docker rmi alfonsodapenacores73/ejercicio3:v3
Untagged: alfonsodapenacores73/ejercicio3:v3
adcorescliente@alfonsocliente:~/ejercicio3$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
alfonsodapenacores73/ejercicio3    v2           54a5b182c052     34 hours ago    453MB
<none>                    <none>       3c9232b8fa53     2 days ago      453MB
alfonsodapenacores73/ejercicio3    v1           af8b68ef914e     2 days ago      453MB
<none>                    <none>       3e5a8a78757b     2 days ago      453MB
<none>                    <none>       8855228582ed     2 days ago      453MB
<none>                    <none>       059917426cdb     2 days ago      453MB
alfonsodapenacores73/myapache2     v2           a54578b71d6a     12 days ago     249MB
<none>                    <none>       ce22a6756d17     12 days ago     207MB
<none>                    <none>       972cab2b6143     12 days ago     207MB
debian                        latest       d84448199470     3 weeks ago     117MB
mariadb                      latest       a914eff5d2eb     8 weeks ago     336MB
alpine                       latest       aded1e1a5b37     8 weeks ago     7.83MB
nginx                        latest       53a18edff809     2 months ago    192MB
hurlenko/filebrowser         latest       6c35bdd7cdb7     2 months ago    24.9MB
phpmyadmin                   latest       052506f2de4d     2 months ago    570MB
redis                        latest       ad4b31aa2de6     3 months ago    117MB
mongo                        4           d896c071ac69     13 months ago   427MB
sdelements/lets-chat         latest       296501fb5b70     8 years ago     255MB
adcorescliente@alfonsocliente:~/ejercicio3$
```

- Una vez borrada la imagen, nos la descargamos de nuestro repositorio remoto a nuestro repositorio local:

```
$ docker pull alfonsodapenacores73/ejercicio3:v3
```

```
adcorescliente@alfonsocliente: ~/ejercicio3
adcorescliente@alfonsocliente:~/ejercicio3$ docker pull alfonsodapenacores73/ejercicio3:v3
v3: Pulling from alfonsodapenacores73/ejercicio3
Digest: sha256:58b5d43fc9ab63785a08fd1011ff2234a7a06105eeef18b3e5b0e9f83587c44a
Status: Downloaded newer image for alfonsodapenacores73/ejercicio3:v3
docker.io/alfonsodapenacores73/ejercicio3:v3
adcorescliente@alfonsocliente:~/ejercicio3$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
alfonsodapenacores73/ejercicio3    v2           54a5b182c052     34 hours ago    453MB
<none>                    <none>       3c9232b8fa53     2 days ago      453MB
alfonsodapenacores73/ejercicio3    v1           af8b68ef914e     2 days ago      453MB
alfonsodapenacores73/ejercicio3    v3           af8b68ef914e     2 days ago      453MB
<none>                    <none>       8855228582ed     2 days ago      453MB
<none>                    <none>       3e5a8a78757b     2 days ago      453MB
<none>                    <none>       059917426cdb     2 days ago      453MB
alfonsodapenacores73/myapache2     v2           a54578b71d6a     12 days ago     249MB
<none>                    <none>       ce22a6756d17     12 days ago     207MB
<none>                    <none>       972cab2b6143     12 days ago     207MB
debian                        latest       d84448199470     3 weeks ago     117MB
mariadb                      latest       a914eff5d2eb     8 weeks ago     336MB
alpine                       latest       aded1e1a5b37     8 weeks ago     7.83MB
nginx                        latest       53a18edff809     2 months ago    192MB
```

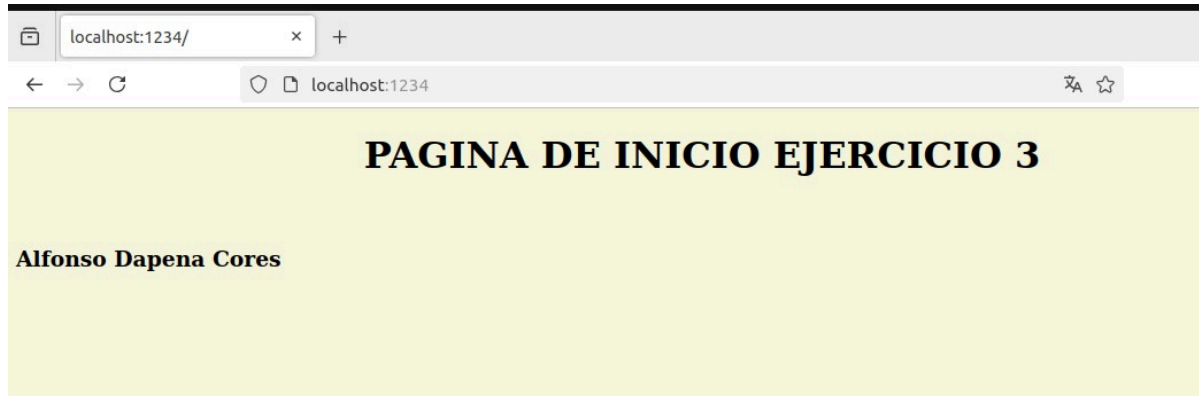
3.6 Creación nuevo contenedor modificando el puerto

- Ahora que hemos descargado la imagen de nuestro repositorio remoto, procedemos a la creación de un nuevo contenedor usando la imagen descargada, pero variando el puerto al 1234. Después, comprobamos que el contenedor se está ejecutando:

```
$docker run -d -p 1234:80 --name ejercicio_3_p_1234
alfonsodapenacores73/ejercicio3:v3
$docker ps
```

```
adcorescliente@alfonsocliente: ~/ejercicio3
adcorescliente@alfonsocliente:~/ejercicio3$ docker run -d -p 1234:80 --name ejercicio_3_p_1234 alfonsoDapenaCores73/ejercicio3:v3
3d7b518562aa50a219f748785c46a706e10f1f4a220fa23623d46aff6047
adcorescliente@alfonsocliente:~/ejercicio3$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
3d7b518562aa   alfonsoDapenaCores73/ejercicio3:v3 "docker-php-entrypoi..." 5 seconds ago  Up 4 seconds  0.0.0.0:1234->80/tcp, [
::]:1234->80/tcp
ejercicio_3_p_1234
```

- A continuación, abriendo nuestro navegador, comprobamos que podemos acceder al fichero `index.html` de nuestro servidor apache en el puerto 1234, introduciendo la siguiente dirección: <http://localhost:1234>



- También comprobaremos que nuestro script `php` funciona, introduciendo la dirección <http://localhost:1234/fecha.php>

